A Research Proposal

on

ANALYSIS OF SHA-3 FINAL ROUND CANDIDATE ALGORITHMS AND DESIGN OF VARIANT TO SKEIN HASH FAMILY

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

In partial fulfillment of the requirements for the award of degree of

DOCTOR OF PHILOSOPHY (Ph.D.) IN COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Rajeev Sobti

Supervised by:

Dr. G. Geetha

LOVELY FACULTY OF TECHNOLOGY AND SCIENCES

LOVELY PROFESSIONAL UNIVERSITY

PUNJAB

1.0 INTRODUCTION (THE PROBLEM AREA TO BE INVESTIGATED)

1.1 Cryptology

Cryptology comprises two complementary fields of research: cryptography and cryptanalysis. A cryptographer is concerned with the development of new schemes or algorithms, to protect the privacy of the information being communicated through any communication channel. A cryptanalyst on the other hand is concerned with the development of attack methodologies that break a cryptographic algorithm, allowing, for example, unauthorised people access to secret information or the ability to forge documents.

Until mid-1970, the term cryptography and cryptanalysis refers to protection and violation of confidentiality of Information respectively. It was believed that if the confidentiality is achieved, then other security goals like authentication can be achieved automatically [1]. It was in year 1976 when Diffie and Hellman in their seminar paper "New Directions in Cryptography" [2] introduced a number of relevant concepts like Digital Signatures and differentiated confidentiality from Authentication.

These days, Security Service Architecture X.800 has defined five categories of services i.e. Authentication, Access Control, Confidentiality, Data Integrity and Non-repudiation [3]. Different mechanisms and tools are being used to achieve these services.

1.2 Cryptographic Hash Functions:

Cryptographic Hash Functions (aka hash functions) is an important tool in cryptography and is used to achieve a number of Security Goals like Message Authentication, Message Integrity, and are also used to implement Digital Signatures (Non-repudiation), Entity Authentication and Digital Steganography.

Cryptographic Hash Function is an algorithm which receives a variable size input message (M) and produces a fixed size output (h) referred to as hash Code.

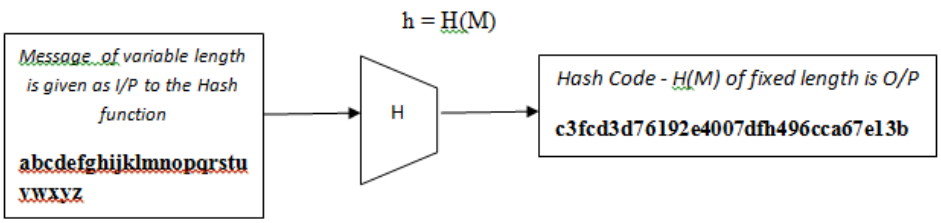
h = H(M)
Message of variable length is given as I/P to the Hash function
abcdefghijklmnopqrstuvwxyz
H
Hash Code - H(M) of fixed length is O/P
c3fcd3d76192e4007dfh496cca67e13b

Fig. 1 Compression with a Cryptographic Hash Function (MD5 used here)

Hash code is also known as message digest, checksum, hash values, hash sums, or simply hashes. The purpose of the hash function is to produce a fingerprint of a file, message or other block of data.

A hash function must possess the following three properties (adapted from [4]):

Pre-Image Resistance For any given value h, it is computationally infeasible to find x such that H(x) = h. This is sometime also referred as one-way property.

Second Pre-Image Resistance For any given block x, it is computationally infeasible to find $y \neq x$ such that H(y) = H(x). This is sometime also referred as Weak Collision resistance.

Collision Resistance It is computationally infeasible to find any pair (x,y) such that H(x) = H(y). This is sometime also referred as strong Collision resistance.

1.3 Applications of Hash Functions:

The field of cryptography may be classified into two sub branches: symmetric key cryptography and asymmetric key cryptography. In the former case, both the communicating parties share the same secret key for achieving security while communicating with each other whereas in the latter case they use two keys: public key and private key. Private Key is kept secret by the communicating party but the public key is known to others. Now it is interesting to see to which branch of cryptology hash functions fit.

Hash functions have wide range of applications based on Symmetric key as well as on Asymmetric keys few of which are highlighted here.

1.3.1 Achieving Integrity & Authentication Verifying the integrity and authenticity of information is a prime necessity in computer systems and networks. In particular, two parties communicating over an insecure channel require a method by which information sent by one party can be validated as authentic (or unmodified) by the other. [5]

Message Integrity & Authentication may be implemented in multiple ways. *Symmetric Encryption based mechanisms* may be used but they have their own drawbacks. Drawbacks like speed, cost factor, optimization for data sizes etc. have been highlighted by Tsudik [6]. Such methods combine the Confidentiality and Authentication functions. However there are scenarios where encrypting full message (confidentiality) is not

required. For such applications keeping message secret is not the concern but authenticating it is important. For example in SNMPv3 (Simple Network Management Protocol version 3), it is usually important for a managed system to authenticate incoming SNMP commands (like changing the parameters at the managed system), but concealing the SNMP traffic is not required.

In order to implement message authentication and integrity, the alternative techniques (other than the methods mention in last paragraph) are *MAC* (Message Authentication Code) or *Cryptographic Hash Functions*. MACs may be constructed out of block ciphers like DES. More recently, however, there has been a surge of interest in the idea of constructing MACs from cryptographic Hash Functions [5]. In addition to using Hash Functions for implementing MAC, Hash functions can be used to achieve message authentication and integrity goals without the use of symmetric encryption. Tsudiac [6] has detailed a protocol based on the same idea. The usage of Hash Functions for Message Authentications and ensuring message integrity has surged because majority of hash functions are faster than block ciphers in software implementation and these software implementations are readily and freely available [5].

1.3.2 Implementing Efficient Digital Signatures Digital signature is a security goal of a cryptosystem which intends to achieve the goal of authenticity and a security service or property of non-repudiation [1]. MAC and Hash Functions alone do not implement the Security goal of Digital Signatures. It was Diffie and Hellman [2] who first realised the need for a message dependent electronic signature (fingerprint) to avoid disputes between sender and receiver. RSA [7] was the first public key crypto systems with digital signature capabilities. However there has been an interesting part of this invention. James Ellis, Clifford Cocks and Malcolm Willaimson from GCHQ (Government Communication Head Quarters), Cheltenham, Britain perhaps invented the idea of Public key in 1972. The three Britons had to sit back and watch as their discoveries were rediscovered by Diffie, Hellman, Merkle, Rivest, Shamir and Adleman over the next three years because of the polices of GCHQ that all work is top secret and cannot be shared with anyone [8].

Hash functions are used to optimize the digital signature schemes. Without the use of Hash, the signature will be of same size as message. The fundamental concept here is instead of generating the signature for the whole message which is to be authenticated;

the sender of the message only signs the digest of the message using a signature generation algorithm. The sender then transmits the message and the signature to the intended receiver. The receiver verifies the signature of the sender by computing the digest of the message using the same hash function as the sender and comparing it with the output of the signature verification algorithm. It is obvious that this approach saves a lot of computational overhead involved in signing and verifying the messages in the absence of hash functions [1].

1.3.3 Authenticate Users Hash functions may be used to authenticate the users at the time of login. The passwords are stored in the form of message digest to avoid access of the same even to Database Administrators (because of Pre-Image resistance of Hash digest). Whenever user tries to login and enter the password, the message digest of the entered password is computed and compared with the digest stored in the database. If it matches, then login is successful, otherwise user is not authenticated.

1.3.4 Digital Time Stamping Majority of text, audio and video documents are available in digital format and a number of simple techniques and tools are available to change digital documents. So some sort of mechanism is required to certify when such a document was created to last modified. One way hash functions and digital signatures can be used to implement the same [9].

In addition to the above, Hash Functions may also be used to index data in hash tables, for fingerprinting, for generating random numbers, to detect duplicate data or uniquely identify files, and as checksums to detect unintentional data corruption also.

Looking at this wide range of applications, it is not correct to say that Hash Functions belong to one particular cryptographic sub branch. These cryptographic tools deserve a separate status for themselves. They are used in almost all places in cryptology where efficient information processing is required.

1.4 Structure of Hash Functions

Merkle [10] and Damgard [11] proposed an iterative structure to construct a collision resistant cryptographic hash function using a fixed length input collision resistant compression function (f).

This iterated Hash function design (also known as Merkle – Damgard Design) has the following structure:

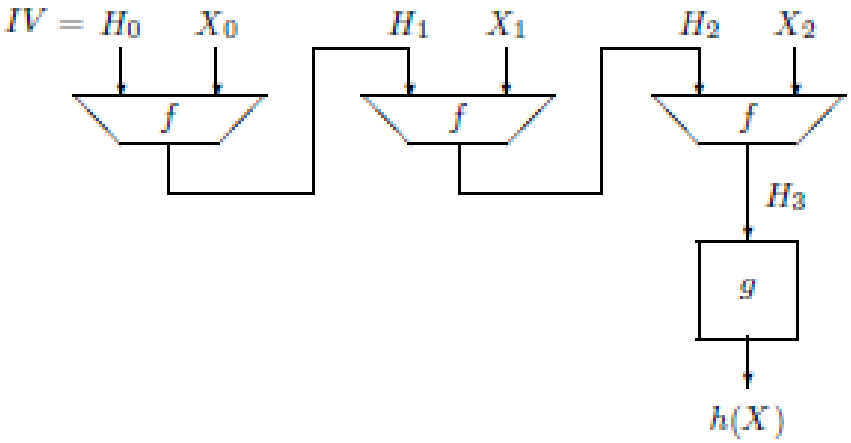

Fig. 2 Example of Iterative Hash Computation [12]

IV – Initial Value, H_i - Chaining Variable, X_i – ith I/P Block, h(X) – Message Digest

F – Compression Function

In above example, algorithm computes the hash result for an input X that is divided into three blocks X0; X1; X2. A compression function f is used which takes two inputs: a chaining variable Hi and a block Xi. The chaining variable input for the first iteration is equal to an initial value IV. For the next iterations the chaining variable input corresponds to the previous compression function output. The result H3 from the last application of the compression function is sent to an output transformation g which computes the hash result h(X) = h(X0||X1||X2) = g(H3).

More generally, for an input X consisting of t blocks X0; X1; : : : X_{t-1} the iterative computation of the hash result can be described as follows:

H_0 = IV ;

$H_{i+1} = f(H_i;X_i)$ for 0 <= i < t ;

$h(X) = g(H_t)$:

Majority of Hash Functions launched in recent years and being used these days follow the iterated hash function. MD4 [13], MD5 [14], SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 [15] all are influenced by the Merkle and Damgard's iterated hash design as explained above.

1.5 Attacks on Hash Functions

Attacks on Hash functions can be classified into two broad categories:

a) Brute Force Attacks

b) Cryptanalysis Attack

Brute Force Attack Brute force attacks work on all hash functions independent of their structure and any other working details. They are similar to exhaustive search or brute-force key recovery attacks on the encryption schemes to extract the secret key of the encryption scheme. The security of any hash function lies in its output bit size. For a hash code of length n, the level of effort required to resist different brute force attacks on hash functions is as follow:

a. **Pre-image attack** --- Effort required for brute force attack = 2^n

In this attack, for a given n-bit digest h of the hash function H( ), the attacker evaluates H( ) with every possible input message M until the attacker obtains the value h.

b. **2^{nd} Pre-image attack** --- Effort required for brute force attack = 2^n

In this attack, for a given message M and the hash function H( ), the attacker tries H( ) with every possible input message $M' \neq M$ until the attacker obtains the value H(M).

c. **Collision attack** --- Effort required for brute force attack = $2^{n/2}$

In this attack, for a given hash function H, the attacker tries to find two messages M and M' such that $M \neq M'$ and H(M) = H(M'). On average the opponent would have to try $2^n / 2$ (= 2^{n-1}) messages to find one that matches the hash code of the intercepted message However a chosen plain text attack (based on Birthday Paradox) is possible and in that case the effort required for collision in a Hash function is $2^{n/2}$ in place of 2^{n-1} [16].

Cryptanalysis Attack Cryptanalysis of Hash functions focuses on the construction of general hash function and on the structure of Compression Function.

Due to fixed size of the hash values compared to much larger size of the messages, collisions must exist in hash functions. However, for the security of the hash function, such collisions

must be computationally infeasible to find. Collisions in hash functions are much easier to find than pre-images or 2^{nd} pre-images.

Informally, a hash function is said to be "broken" when a reduced number of evaluations of the hash function compared to the brute force attack complexities and the strengths estimated by the designer of the hash function are used to violate at least one of its properties immaterial of the computational feasibility of that effort. For example, assume that it requires 2^{90} evaluations of the hash function to find a collision for a 256-bit hash function. Though it is impractical to generate this amount of computational power today, the hash function is said to be broken as this factor is less than the 2^{128} evaluations of the hash function required by the Birthday attack.

It should be noted that hash functions are easier to attack practically than encryption schemes because the attacker does not need to assume any secrets and the maximum computational effort required to attack the hash function is only upper bounded by the attacker's resources not users gullibility. This is not the case with block ciphers where the maximum practical count of executions of the block algorithm is limited by how much computational effort the attacker can get the user to do [1].

Gauravaram [1] in his Ph.D. thesis has classified attacks on hash function in two categories i.e. Generic and Specific attacks. The attacks that work on a general hash function construction are called generic attacks. For example, attacks on the Merkle-Damgard construction that work on all hash functions designed using it are the generic attacks on the Merkle- Damgard hash functions. The attacks that work on specific hash functions designed following a particular hash function design framework are called specific attacks. For example, collision attacks on the specific hash functions MD4 [17], MD5 [18,19], SHA-0 [20,21] and SHA-1 [20, 22] that follow the principles of Merkle-Damgard iterative structure come under the category of specific attacks [1].

2.0 REVIEW OF LITERATURE

In all hash functions input is viewed as a sequence of n bit blocks. The input is processed one block at a time in an iterative fashion to produce an n-bit hash function. Over the years a number of techniques have been devised to process the blocks in a manner so that its structure cannot be exploited for any cryptanalytic attack.

The simplest technique can be bit by bit XOR of every block or one bit circular shift or rotation can be performed on hash value after each block is processed. But such techniques are not helpful. Given a message, it is an easy matter to produce new message that yields that hash code. One just need to prepare a desired fraudulent message and then append n-bit blocks that forces the fraudulent message plus block to yield the same hash code as for original message.

Generating hash functions using Cipher Block chaining technique without a secret key were also studied by **Rabin in 1978** [3]. It uses symmetric encryption like DES to compute the hash code as follow. M – Message is divided into fixed size N blocks and then DES is used:

$$H_0 = \text{initial value}; H_i = E(M_i, H_{i-1}), \text{Hash Code} = H_N$$

But such techniques were prone cryptanalysis using to meet-in-the middle attack a variations of Birthday attack. As the encryption algorithm used was DES (i.e. hash code of 64 bit) so brute force attack is also relatively easy.

Based on study of **Davies and Price in 1985** and **Matyas and Meyer in 1985**, Gauravram [1] in his Ph.D. thesis described variation in the above scheme known as Davies – Meyer Compression Function and in this feed-forward was used i.e. $H_i = E(M_i, H_{i-1}) + H_{i-1}$.

where + could be any group operation and these hash functions use modular addition as the group operation in the Davies-Meyer compression function. Gauravaram [1] in his Ph.D thesis also highlighted the study of **Miyaguchi in 1990** that these compression functions are vulnerable against the fixed point attacks where a pair (H_{i-1}, M_i) can be found such that $H_{i-1} = f(H_{i-1}, M_i)$. There exists one *and only one* fixed point for every message block for these compression functions. The attacker finds a fixed point for the compression function f by choosing any message block M_i and inverting the function g as given by $g^{-1}(0, M_i)$ to get a state H_{i-1} such that $f(H_{i-1}, M_i) = H_{i-1}$. This costs just one

inverse operation which is equivalent to the computation of one compression function iteration.

Juneman in 1987 [23] showed that some form of Birthday attack will succeed against any hash scheme involving the use of cipher block chaining without a secret key provided that either the resulting hash code is small enough (e.g. 64 bits or less) or that a larger hash code can be decomposed into independent sub-codes.

Merkle [10] **and Damgard [11] in 1989** independently gave the motivation for iterative hash structure. The same iterative hash Structure is discussed in the introduction section. The motivation for such an iterative hash function consisting of fixed input Compression function stems from the observation by Merkle [10] and Damgard [11] that if the compression function is collision resistant, then so the resultant iterated hash function. So the problem of designing secure hash function reduces to that of designing a collision resistant Compression Function that operates on input of some fixed size. Most popular of today's Hash function including MD5 [15], SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 [29] the standard hash functions given by NIST are following this structure.

In 1990, **Rivest** designed Message Digest 4 (**MD4**) [13]. **In 1991, MD5** [14] was designed by **Rivest** to replace the earlier Hash MD4. It became a milestone in the development of Hash. It was a widely-used well-known 128-bit iterated hash function, used in various applications including SSL/TLS, IPsec, and many other cryptographic protocols. It was also commonly-used in implementations of time stamping mechanisms, commitment schemes, and integrity-checking applications for online software and random-number generation.

In 1993, Secure Hash Algorithm (SHA) was developed by the National Institute of Standards and Technology (NIST) and published the same as Federal Information Processing Standard (FIPS 180) in 1993. A revised version was issued as FIPS180-1 in 1995 and is generally referred to as SHA-1 [24]. When revised version of SHA-1 was published no details of the weaknesses found in SHA-0 were provided [20]. SHA-1 produces a hash value of 160 bit.

In **1993 B. den Boer and A. Bosselaers** [25] gave a limited result of finding a "pseudo-collision" (identical digest produced with two different initialization vectors) of the MD5 compression function.

In 1996 H. Dobbertin [26] published an attack, without details, that found a collision in MD5 with an IV (Initial value) chosen by him that was different from the one actually used in MD5 . While this was not an attack on the full MD5 hash function, but it was close enough to initiate a debate in the cryptographic community and in turn propose a a replacement, like SHA-1.

In 2002, NIST produces a revised version of the standard known as FIPS180-2 [27] and that defined three new versions of SHA with Hash Value lengths of 256, 384 and 512 and knows as SHA-256, SHA-384, and SHA-512. So total SHA versions becomes four including SHA-1 (160 bit)

In year 2004, at CRYPTO'04, a team of researchers from the **Shandong University in Jinan China, led by Xiaoyun Wang**, announced collisions in MD5 as well as collisions in a host of other hash functions including MD4, RIPEMD, and HAVAL-128 [18]. Wang along with his team were able to find many real collisions which are composed of two 1024-bit messages with the original initial value IV of MD5. This Analytical attack was reported to take only one hour on an IBM p690 cluster. Moreover, their attack works for any given initial value. Their findings were published at EUROCRYPT in 2005 [17, 19].

In year 2005, **Biham *et al.*** [20] presented a tool that uses near-collisions in order to find collisions using a multi-block technique. This tool can be used to attack variants that cannot be attacked by the original technique proposed by the same team a year back, as well as to reduce complexities of attacking other variants. With some additional refinements, it also improves the attack on full SHA-0, reducing the complexity down to 2^{51}. Author also presented attacks on reduced-round SHA-1, which can find collisions of up to 53–58 rounds faster than the birthday attack, and show new techniques to attack SHA-1.

In year 2005, NIST announced the intention to phase out approval of SHA-1 and move to rely on other SHA versions by 2010 [3]. Shortly thereafter the **Wang *et al.*** presented two papers at the 2005 CRYPTO conference [21, 22] detailing their attacks on SHA0 and SHA1. The team was able to find collisions of the full 80-step SHA-0 with complexity less than 2^{39} hash operations and team showed that collision in SHA-1 can be found with complexity less than 2^{69} hash operations. This was the first attack on the full 80-step SHA-1 with complexity less than the 280 theoretical bound.

Given the variety of hash functions attacked by this team, it seemed likely that their approach may prove effective against all cryptographic hashes in the MD family, including all variants of SHA [28].

Burr from US National Institute of Standards and Technology [29] in his paper reviewed the scenarios of Cryptographic Hash Functions. Burr [29] pointed out that "with SHA-1 and SHA-2 in its cryptographic toolkit, NIST had hoped to be done with hash functions for a long time. Aside from a near break of MD5 by Dobbertin [26] in 1996, researchers made little progress in hash function analysis until mid-2004. Since then, Wang, Antonine Joux, and Eli Biham have attacked nearly all the early hash functions, including SHA-1. Given that SHA-2 functions are in the same family as the earlier broken functions, these attacks shook cryptographers' long term confidence in nearly all hash functions designed to date. Cryptographers have learned much about hash functions and how to attack them in the past couple of years, and yet cryptanalysts generally agreed that practical attacks on the SHA-2 hash functions are unlikely in the next decade. However, attacks and research results could reduce their strength well below theoretical work levels (2^{112}, 2^{128}, 2^{192}, and 2^{256} operations for SHA-224, SHA-256, SHA-384, and SHA-512, respectively)."

Hoch and Shamir in year 2006 [30], studied the multi collisions on Iterated Concatenated Expanded (ICE) Hash Functions. Hoch and Shamir extended the idea presented by **Joux in 2004** [31] **. Joux in 2004** [31] showed that " in any iterated hash function it is relatively easy to find exponential sized multicollisions, and thus the concatenation of several hash functions does not increase their security". But **Joux** [31] Attack does not work on ICE i.e. when in addition to Iterated and Concatenated Hash Function technique message Expansion is also added i.e. each iterated function process message block more than once. **Hoch *et al.* [30]** considered the general case (ICE) and proved that even if we allow each iterated hash function to scan the input several times in random expanded order, their concatenation is not stronger than a single function. Finally, authors extended their result to tree-based hash functions with arbitrary tree structures. Hoch *et al.* showed that a large class of natural hash functions (ICE and its generalization TCE) is vulnerable to a multicollision attack, and hoped that the techniques developed here will help in creating multicollision attacks against even more complicated types of

hash functions. Such a conclusion was perhaps hinting to probable attack on SHA 2 family of hash functions.

Looking at the current scenarios, In **Nov 2007** NIST (National Institute of Standards and Technology) announced a **public competition** [32] to develop a new cryptographic hash algorithm to replace the older SHA-1 and SHA-2. The competition was NIST's response to advances in the cryptanalysis of hash algorithms. The winning algorithm will be named "SHA-3", and will augment the hash algorithms currently specified in the Federal Information Processing Standard (FIPS) 180-3, Secure Hash Standard [15]. As per NIST website "NIST is initiating an effort to develop one or more additional hash algorithms through a public competition, similar to the development process for the Advanced Encryption Standard (AES)." [32]

By October 31, 2008, NIST received **sixty-four** entries; and **selected fifty-one** candidate algorithms to advance to the **first round on December 10, 2008**, and fourteen advanced to the **second round on July 24, 2009**. A year was allocated for the public review of the fourteen second-round candidates. NIST received significant feedback from the cryptographic community. Based on the public feedback and internal reviews of the second-round candidates, NIST selected five SHA-3 finalists – **BLAKE [33], Grøstl [34], JH [35], Keccak [36],** and **Skein [37]** to advance to the third (and final) round of the competition **on December 9, 2010**, which ended the second round of the competition. A one-year public comment period is planned for the finalists. NIST also plans to host a final SHA-3 Candidate Conference in the spring of 2012 to discuss the public feedback on these candidates, and select the SHA-3 winner later in 2012 [32].

3.0 OBJECTIVES /SCOPE OF THE STUDY

There are two aspects of my study as detailed below:

a) Analysis of SHA-3 Final round candidate algorithms

Motivation:

For SHA-3 competition, NIST announced the "Reference Platform and Complier" [32] on which the candidate algorithms will be evaluated. Reference platform and compiler, as announced by NIST is Wintel personal computer, with an Intel Core 2 Duo Processor, 2.4GHz clock speed, 2GB RAM, running Windows Vista Ultimate 32-bit (x86) and 64-bit (x64) Edition and ANSI C compiler in the Microsoft Visual Studio 2005 Professional Edition. Because of this reference platform, there is high probability that final round candidates have optimized their code for the reference platform and compiler.

But a considerable domain of architecture like 8 bit or 16 bit architectures used by Smart Cards, clusters, embedded systems etc. may have skipped the attention of cryptographers of final round. It may happen that a seemingly fast function on reference platform is not performing well on embedded system like mobile phones. **So an analysis of SHA-3 final round candidate algorithms is desired**. In fact NIST notice introducing the competition [32] also invites the public to conduct similar tests and compare results on additional platforms

Goal

Research work will comprise of analysis of SHA-3 final round candidate algorithms (i.e. – BLAKE [33], Grøstl [34], JH [35], Keccak [36], Skein [37]) for their underlying structure, their Implementation and performance comparisons on any one of the platforms other than the one specified as reference Platform by NIST. The platform may be ARM processors, 8-bit/16 bit machine, various languages & compilers like C or Java.

b) Design of Variant to Skein Hash Function

Motivation:

Skein (one of the finalists of SHA-3) as submitted by Schneier and his team [37] is a family of hash functions with three different internal state sizes: 256, 512, and 1024 bits and can support any output size.

Skein's idea is to build a hash function out of a tweakable block cipher. Threefish is the tweakable block cipher at the core of Skein, and defined with a 256,512, and 1024-bit block size. Tweakable Block ciphers have not only the usual inputs - message and cryptographic key, but also a third input, the **tweak**." The tweak serves much the same purpose that an initialization vector does for CBC [38]. The use of a tweakable block cipher allows Skein to hash configuration data along with the input text in every block, and make every instance of the compression function unique. This property directly addresses many attacks on hash functions, and greatly improves Skein's flexibility.

In October 2010, an attack that combines rotational cryptanalysis with the rebound attack was published by Khovratovich and his team. The attack finds rotational collisions for 53 of 72 rounds in Skein-256, and 57 of 72 rounds in Skein-512. It affects the Threefish cipher [39]. This is a follow-up to the earlier attack published in February, which breaks 39 and 42 rounds respectively [40]. So **Skein is prone to Rotational Cryptanalysis and an alternative is desired**.

Goal:

Design a variant of Skein Hash function, to be more precise replace the Threefish Block cipher (the core building block of Skein Hash Function) with the new algorithm (to be developed) so that it is more secure and is lesser prone to Rotational Cryptanalysis.

4.0 PROPOSED METHODOLOGY

To achieve the objective as mentioned above, the following methodology is proposed:

a) Analysis of SHA-3 final round candidate algorithms

The first step will be to decide the target Architectures and compliers on which various SHA-3 final round candidate algorithms is to be evaluated. SHA-3 final round candidate algorithms will be re-implemented from scratch in the Language & compiler chosen in first step. The code (in C language) as provided with submission packages will not be used. Similar optimization techniques and efforts will be applied to each algorithm so that resulting performance differences may then be considered as inherent to the algorithms themselves. The code will be benchmarked on a variety of platforms as decided in the first step and then conclusions may be drawn on the performance of SHA-3 final round candidate algorithms and on general parameters that affect the performance of hash functions.

b) Design of Variant to Skein Hash Family

The Threefish algorithm is based on Modular Arithmetic, Rotations and XOR operations. The rotational cryptanalysis as presented by Khovratovich and his team [39, 40] is based on the analysis that a rotational pair of inputs is converted to a rotational pair of outputs with a probability depending only on the number of additions in the scheme. Hence, there is universal upper bound on the security of the ARX primitives and so of Threefish ciphers.

To overcome the same, some new Boolean function has to be thought of and has to be designed that act as the building block for the block cipher to be used within Skein in place of Threefish. Boolean function may be designed using addition modulo 2^n, Multiplication with constant, XOR or any bit wise Boolean functions. The main factors that will be kept in mind while designing the Boolean function may be: Balancedness, Non linearity, Algebric Degree and Correlation Immunity.

Once the Boolean function with desired objectives has been formulated, the code for the hash function will be written and the same has to be tested for rotational, differential and linear cryptanalysis. To test some sample data is to be generated and then for testing either code may have to be written or the existing framework like the one provided by Mouha, Velichkov, Canni`ere and Preneel [41] may be used.

To validate the new algorithm to be developed, it will be published on Cryptology eprint Archive and kept open for analysis by cryptology community.

5.0 REFERENCES

[1] P. Gauravram, "Cryptographic Hash Functions: Cryptanalysis, design and Applications," Ph.D. Thesis, Queensland University of Technology, Brisbane, Australia, 2007

[2] W. Diffie, M. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, vol. 22(6), pp. 644-654, Nov. 1976.

[3] W. Stallings, *Cryptography and Network Security*", 4th ed., Pearson Prentice Hall, 2009.

[4] P. Rogaway, and T. Shrimpton, "Cryptographic hash-function basics: Definitions, implications and separations for preimage resistance, second preimage resistance, and collision resistance", *in Proc. FSE*, 2004, pp.371-388.

[5] M. Bellare, R. Canetti and H. Krawczyk, "Keying Hash Functions for Message Authentication", *in Proc. CRYPTO'96*, 1996, pp.1-15.

[6] G. Tsudik, "Message Authentication with One-Way Hash Functions", *in Proc.* INFOCOM, 1992, pp.2055-2059

[7] R.L. Rivest, A. Shamir, and L.M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *presented at Commun. ACM*, 1978, pp.120-126

[8] S. Singh. (1999) The Alternative History of Public-Key Cryptography. [Online]. Available: http://cryptome.org/ukpk-alt.htm

[9] S. Haber and W. Stornetta, "How to time-stamp a digital document", *Journal of Cryptology*, vol. 3(2), pp. 99-111, 1991.

[10] R.C. Merkle, "One Way Hash Functions and DES", *in Proc. CRYPTO*, 1989, pp.428-446.

[11] I. Damgård, "A Design Principle for Hash Functions", *in Proc. CRYPTO*, 1989, pp.416-427.

[12] B. Rompay, “Analysis and design of cryptographic Hash functions, MAC algorithms and Block ciphers”, Ph.D. Thesis, Katholieke Universiteit Leuven, Belgium, 2004.

[13] R. Rivest, *The MD4 message digest algorithm*, IETF RFC 1320, April 1992; www.rfc-editor.org/rfc/rfc1320.txt.

[14] R. Rivest, *The MD5 message digest algorithm*, IETF RFC 1321, April, 1992; www.rfc-editor.org/rfc/rfc1320.txt.

[15] *FIPS Std. FIPS180-3, Secure Hash Standard (SHS)*, NIST, 2008.

[16] M. Bellare and T. Kohno, "Hash Function Balance and Its Impact on Birthday Attacks", *in Proc. EUROCRYPT*, 2004, pp.401-418.

[17] X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu, "Cryptanalysis of the Hash Functions MD4 and RIPEMD", *in Proc. EUROCRYPT*, 2005, pp.1-18.

[18] X.Wang, D. Feng, X. Lai, and H. Yu, “Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD," *Cryptology ePrint Archive*, Report 2004/199, 2004.

[19] X. Wang and H. Yu, "How to Break MD5 and Other Hash Functions", *in Proc. EUROCRYPT*, 2005, pp.19-35.

[20] E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet, and W. Jalby, "Collisions of SHA-0 and Reduced SHA-1", *in Proc. EUROCRYPT*, 2005, pp.36-57.

[21] X. Wang, H. Yu, and Y. L. Yin, "Efficient Collision Search Attacks on SHA-0", *in Proc. CRYPTO*, 2005, pp.1-16.

[22] X. Wang, Y. L. Yin, and H. Yu, "Finding Collisions in the Full SHA-1", *in Proc. CRYPTO*, 2005, pp.17-36.

[23] R. Juneman, “Electronic Document Authentication”, *IEEE Network Magazine*, April 1987.

[24] *FIPS Std. FIPS180-1, Secure Hash Standard (SHS)*, NIST, 1995.

[25] B. D. Boer and A. Bosselaers, “Collisions for the Compression function of MD5”, *in Proc. EUROCRYPT*, 1993, pp. 293 - 304.

[26] H. Dobbertin, “Cryptanalysis of MD5 compress”, *in Proc. EUROCRYPT*, 1996

[27] *FIPS Std. FIPS180-2, Secure Hash Standard (SHS)*, NIST, 2002.

[28] J. Black, M. Cochran, and T. Highland, "A Study of the MD5 Attacks: Insights and Improvements", *in Proc. FSE*, 2006, pp.262-277.

[29] W. E. Burr, "Cryptographic Hash Standards: Where Do We Go from Here", *IEEE Security & Privacy*, vol. 4(2), pp.88-91, March 2006.

[30] J. J. Hoch, A. Shamir, "Breaking the ICE - Finding Multicollisions in Iterated Concatenated and Expanded (ICE) Hash Functions", *in Proc. FSE*, 2006, pp.179-194.

[31] A. Joux, "Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions", *in Proc. CRYPTO*, 2004, pp.306-316.

[32] Computer Security Resource Center, National Institute of Standards and Technology website. [Online] Available: http://csrc.nist.gov/groups/ST/hash/index.html

[33] J. P. Aumasson, L. Henzen, W. Meier, and R. C. W. Phan, "SHA-3 proposal BLAKE", Version 1.3, December 16, 2010.

[34] P. Gauravaram, L. R. Knudsen, K . Matusiewicz, F. Mendel, C. Rechberger, M. Schlaffer, and S. S. Thomsen, "Grøstl - a SHA-3 candidate", Version 2.0.1, March 2, 2011.

[35] H. Wu, "The Hash Function JH", Version JH42, January16, 2011

[36] G. Bertoni, J. Daemen, M. Peeters' and G. V. Assche, "The Keccak Reference", Version 3.0, January, 14, 2011.

[37] B. Schneier, N. Ferguson, S. Lucks, D. Whiting, M. Bellare, T. Kohno, J. Walker, and J. Callas, "The Skein Hash Function Family", Version 1.3, Oct 1, 2010.

[38] M. Liskov, R.L. Rivest, and D. Wagner, "Tweakable Block Ciphers", *in Proc. CRYPTO*, 2002, pp.31-46.

[39] D. Khovratovich, I. Nikolić I, and C. Rechberger, "Rotational Rebound Attacks on Reduced Skein**",** *Cryptology ePrint Archive*, Report 2010/538, 2010.

[40] D. Khovratovich and I. Nikolic, "Rotational Cryptanalysis of ARX", *in Proc. FSE*, 2010, pp.333-346.

[41] N. Mouha, V. Velichkov, C. Canni`ere, and B. Preneel, "The Differential Analysis of S-functions", *in Proc. SAC'10*, 2010.